TD/TP 3 - Tri à bulles

I. Principe du tri à bulles. Pour trier un tableau, on le parcourt en inversant un élément avec son successeur s'il lui est supérieur. On recommence ensuite le parcours jusqu'à ce que l'on ne rencontre plus de permutations.

Q.1) - Dérouler cet algorithme sur le tableau | 6 | 4 | 1 | 9 | 5 |

Q.2) - Que peut-on dire après un premier parcours du tableau ?

Q.3) - Étant donné un tableau `T` de taille au moins `n`, écrire un algorithme `parcours(T,n)` qui effectue un parcours de gauche à droite des `n` premiers éléments et renvoie vrai s'il y a eu un échange, faux sinon.

Q.4) - Écrire un algorithme `triABulles(T)` qui effectue le tri à bulles du tableau `T`.

Q.5) - Démontrer que cet algorithme trie le tableau donné en entrée.

Q.6) - Calculer la complexité de ce tri.

II. EN TP

On se placera dans le même projet que pour le TP précédent afin d'utiliser la classe **`Lectcolonne`**.

Q.7) - Dans une nouvelle classe **`Tris`**, écrire la méthode `public static` **`void`** `triABulles(`**`double[]`** `T)` qui trie le tableau en utilisant l'algorithme du tri à bulles.

Q.8) - Dans la méthode `main`, tester ce tri sur une colonne du fichier `iris.txt`. Vérifier le résultat en le comparant avec celui obtenu avec `java.util.Arrays.sort`.

Q.9) - En utilisant la méthode `java.lang.Math.random()`, écrire une méthode `public static` **`double[]`** `tableauAleatoire(`**`int`** `n)` qui retourne un tableau aléatoire de `double` entre 0 et 1 de taille `n`.

Q.10) - Tester le tri à bulles sur un tableau de taille `10000`.

Q.11) - En utilisant la méthode `java.lang.System.currentTimeMillis()`, comparer sur des exemples de différentes tailles la vitesse du tri à bulles et du tri utilisé par `java.util.Arrays.sort`.